Using Software Modules

See the commands described below to learn how to use modules.

Note: Software programs on NAS systems are managed either as modules or packages. For information about using software programs that are managed as packages, see Using Software in the NetBSD Packages Collection.

Using Modules

To find out what software modules are available on NAS systems, run the **module avail** command. You can also view a short description of every module by running **module whatis,** or find information about a specific module by running **module help *modulename***.

To use the software provided in a module, you must first load the module by running the **module load** command. You can load several modules at the same time, as follows:

```
% module load module_name_1 module_name_2 ... module_name_N
```

Testing Modules

New software modules and new versions of existing modules are sometimes available for testing before they are released for general use. To access the test modules, run:

```
module use -a /nasa/modulefiles/testing
module avail
module load module_name
```

If you use the test modules, please send your feedback to support@nas.nasa.gov. Let us know whether the modules worked well or, if you had any problems, describe any steps you took to reproduce them.

Note: Test modules may be moved into production or deleted without notice.

Additional Module Commands

Some useful module commands include:

module list
: Lists which modules are already loaded in your environment.

module purge
: Removes all the modules that are loaded in your environment.

module switch *old_module_name new_module_name*
: Enables you to switch between two modules.

module show *module_name*
: Shows changes that will occur in your environment if you load *module_name.*

For more information, see **man module**.

TIP: If the module commands don't work, ensure that the following line is included in your **.cshrc** or **.profile** file:

- For csh: **source /usr/local/lib/global.cshrc**

- For bash: **`source /usr/local/lib/global.profile`**

Creating Your Own Modulefiles

If you maintain software for your own use or to share with your group, it may be useful to create modulefiles that can be loaded, purged, and so on, just like any other modulefile.

To create your own modulefile, you can simply copy an existing modulefile to your directory (commonly named **`/u/`*`your_username`*`/privatemodules`**), and edit it as appropriate.

First, find a module that you want to mimic and run **`module show `*`module_name`***. The modulefile itself will be listed at the top of the **`module show`** output. You can find most available modulefiles in the **`/nasa/modulefiles/toss3`** directory.

Here are some common features in well-designed modulefiles:

```
#%Module
##
##  All modulefiles must begin on the first line with those 8 characters
##  Here, in the comments, you can describe how the package was built, or, better yet,
##  put it in the ModulesHelp section below

proc ModulesHelp { } {
   puts stderr "Put the information you want to display when the user
   does module help on this"
   puts stderr "modulefile. Note each continuation line begins with puts
   stderr and comments in double quotes"
}

# include the following line only if you want to log the loading of the
# modulefile in our system logs
# most users would leave this out
source /nasa/modulefiles/common/log_module_usage.tcl

# if there might be several versions of this software, it would be good to
#set the version number
set version 1.0-something

# setting basepath or SWDIR is for naming the root directory where your
# various bin, include, lib directories reside
# see later in the modulefile on how it is used
set basepath /u/your_userid/dir_name/dir_name

# if this software requires other modules to be loaded first,
# then use the prereq specifier, for example:
prereq comp-intel

# if your software requires other software that you build,
#then you might simply load them, as in:
module load  my_other_module1  my_other_module2

# if this software conflicts with another software that the user
# may inadvertently load, then use:
conflict another_modulefile

# Finally, the actual setting of the PATHs etc.  You have a choice of either
# prepend-path or append-path:
prepend-path   PATH                          $basepath/$version/bin
prepend-path   CPATH                         $basepath/$version/include
prepend-path   FPATH                         $basepath/$version/include
prepend-path   LD_LIBRARY_PATH               $basepath/$version/lib
prepend-path   LIBRARY_PATH                  $basepath/$version/lib
prepend-path   MANPATH                       $basepath/$version/share/man
```

If you plan to share this software with other users in your group, make sure that they have **`read`** permission to every file and directory, and **`execute`** permission on all directories and the executables in the software's **`bin`** directory.

In order to see your newly created modulefile in the output of the **`module avail`** command, run the **`module use`** command first, as follows:

```
% module use -a /u/your_username/privatemodules
```

Note: The **`-a`** option puts the new modulefile at the end of **`module avail`** output. Without it, the modulefile will be listed at the top.

Article ID: 115
Last updated: 20 Dec, 2021
Revision: 34
Filesystems & Software -> Software -> Using Software Modules
https://www.nas.nasa.gov/hecc/support/kb/entry/115/